

A Case Study: Novel Group Interactions through Introductory Computational Physics

Michael J. Obsniuk,¹ Paul W. Irving,^{1,2} and Marcos D. Caballero^{1,2}

¹*Department of Physics and Astronomy, Michigan State University, East Lansing, MI 48824*

²*CREATE for STEM Institute, Michigan State University, East Lansing, MI 48824*

With the advent of high-level programming languages capable of quickly rendering three-dimensional simulations, the inclusion of computers as a learning tool in the classroom has become more prevalent. Although work has begun to study the patterns seen in implementing and assessing computation in introductory physics, more insight is needed to understand the observed effects of blending computation with physics in a group setting. In a newly adopted format of introductory calculus-based mechanics, called Projects and Practices in Physics, groups of students work on short modeling projects – which make use of a novel inquiry-based approach – to develop their understanding of both physics content and practice. Preliminary analyses of observational data of groups engaging with computation, coupled with synchronized computer screencast, has revealed a unique group interaction afforded by the practices specific to computational physics – problem debugging.

PACS numbers: 01.40.Fk, 01.50.H-

Keywords: Physics Education Research, Computation, Debugging, Case Study

I. INTRODUCTION

Since the development of reasonably affordable and fast computers, educators have argued for their inclusion in the classroom as both a learning aid and a tool [1, 2]. With the recent development of high-level programming languages capable of quickly rendering three-dimensional real-world simulations, the argument for the inclusion of computers in the classroom has not only persisted, but has grown [3]. Well into the 21st century, with its prevalence in modern physics research, we see computation being increasingly referred to as the “third pillar” of physics – along with the more canonical pillars of theoretical and experimental physics [4].

One physics curriculum that includes a computational element is Matter & Interactions (M&I). The M&I curriculum differs from a traditional one not only through the inclusion of computation (VPython), but also through its emphasis on fundamental physics principles and the addition of a microscopic view of matter [5, 6]. Recent work [7, 8] involving students’ use of VPython with the M&I curriculum has begun to analyze the patterns seen in implementing and assessing the use of computation in introductory physics courses.

However, numerous questions remain unanswered regarding the processes observed while groups of students work together to model real world phenomena computationally. We extend our research to a novel implementation of M&I with an emphasis on computation in a group setting, called Projects and Practices in Physics (P³), where students negotiate meaning in small groups, develop a shared vision for their group’s approach, and employ science practices to navigate complex physics problems successfully. Borrowing from the rich literature of computer science education research [9, 10], we use the notion of computational debugging in a physics context to help uncover the salient practices unique to computational physics problems.

In this paper, we present a case study of a group of students immersed in this P³ environment solving a computational problem. This problem requires the translation of a number of fundamental physics principles into computer code and vice versa. Our analysis consists of qualitative observations in an attempt to describe, rather than generalize, the computational interactions, debugging strategies, and learning opportunities bolstered by this novel environment. We have observed two distinct strategies well suited to computational tasks, demonstrating the benefit of the inclusion of this modern tool.

II. ASSUMPTIONS

In an increasingly technological world, we have at our disposal computers well suited for the most procedural and tedious, yet indispensable of STEM tasks. Accordingly, in P³, we treat the computer as an indispensable *modern tool* with which students must familiarize themselves. In spite of the fact that modern computers *take* meaningful direction quite well, they do not yet possess the faculty to *generate* meaningful direction on their own. This necessitates a human factor, in which groups of students must leverage their understanding of fundamental physical principles to model real world phenomena computationally (i.e., generate meaningful direction to a computer).

We focus our research on the social exchanges between group members and the interactions between group and computer, illustrated in Fig. 1, to begin to understand the ways in which computation can influence learning. Particularly, we are interested in the interactions occurring simultaneously with social exchanges of fundamental physics principles (FPPs) specific to the present task (e.g., discussing $d\mathbf{r} = \mathbf{v} dt$ on a motion task) and the display of desirable strategies (e.g., divide-and-conquer). These group-computer interactions vary in form, from ac-

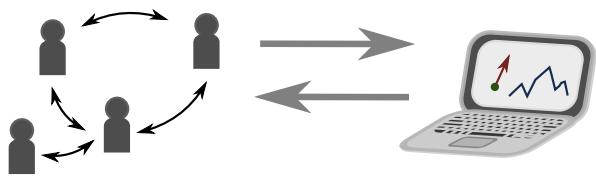


FIG. 1. A group of students interacting with VPython where social exchanges focus on FPPs and desirable strategies are observed through discussion and action.

tively sifting through lines of code, to observing a three-dimensional visual display.

One previously defined computational interaction that reinforces desirable strategies [10], borrowing from computer science education research, is the process of debugging. Computer science [9] defines debugging as a process that comes after testing *syntactically* correct code where programmers “find out exactly where the error is and how to fix it.” Given the generic nature of the application of computation in computer science environments (e.g., data sorting, poker statistics, or “Hello, World!” tasks), we expect to see different strategies more aligned with the goals of a computational *physics* environment. Thus, we extend this notion of computer science debugging into a physics context to help uncover the strategies employed while groups of students debug *fundamentally* correct code that produces unexpected physical results.

III. DATA

In Fall 2014, P³ was run at Michigan State University in the Physics Department. It was this first semester where we collected *in situ* data using three sets of video camera, microphone, and laptop with screencasting software to document three different groups each week. From the subset of this data containing computational problems, we *purposefully sampled* a particularly interesting group in terms of their computational interactions, as identified by their instructor. That is, we chose our case study not based on generalizability, but rather on the group’s receptive and engaging nature with the project as an *extreme case* [11].

The project that the selected group worked on for this study consists of creating a computational model to simulate the geostationary orbit of a satellite around Earth. In order to generate a simulation that produced the desired output, the group had to incorporate a position dependent Newtonian gravitational force and the update of momentum, using realistic numerical values. The appropriate numerical values are Googleable, though instructors encouraged groups to solve for them analytically.

This study focuses on one group in the fourth week of class (the fourth computational problem seen) consisting of four individuals: Students A, B, C, and D. The group had primary interaction with one assigned instructor. Broadly, we see a 50/50 split on gender, with one

ESL international student. Student A had the most programming experience out of the group. It is through the audiovisual and screencast documentation of this group’s interaction with each other and with the technology available that we began our analysis. The majority of the inferences made are based on the audio of discussion, while supporting evidence is gathered from the actions (e.g., writing equations or gesturing) observed through video.

IV. ANALYSIS

To focus in on the group’s successful physics debugging occurring over the 2 h class period, we needed to identify phases in time when the group had recognized and resolved a physics bug. These two phases in time, *bug recognition* and *bug resolution* are the necessary limits on either side of the process of *physics debugging*. We identified these two bounding phases at around (60 ± 5) min into the problem, and further examined the process of debugging in-between. That is, we focused on the crucial moments surrounding the final modifications that took the code from producing unexpected output to expected output.

A. Bug recognition

At around 55 min into the problem, following an intervention from their instructor, the group began to indicate that they were at an impasse:

SB: We’re stuck.
SD: Yeah...

The simulation clearly displayed the trajectory of the satellite falling into the Earth – not the geostationary orbit they expected – as observed on the screencast. This impasse was matched with an indication that they believed the FPPs necessary to model this real world phenomenon were incorporated successfully into the code:

SB: And it’s gonna be something really dumb too.
SA: That’s the thing like, I don’t think it’s a problem with our understanding of physics, it’s a problem with our understanding of Python.

Instead of attributing the unexpected output with a mistake in their understanding or encoding of FPPs, they instead seemed to place blame on the computational aspect of the task.

During this initial phase, we see a clear indication that the group has recognized a bug – there is an unidentified error in the code, which must be found and fixed:

SA: I don’t know what needs to change here...
SD: I mean, that means we could have like anything wrong really.

Although they have identified the existence of the bug, they still are not sure how to fix it – this necessitates the process of debugging.

B. Physics debugging

Within the previously identified phase of bug recognition, the group developed a clear and primary task: figure out exactly how to remove the bug. Eventually, following a little off-topic discussion, the group accepted that in order to produce a simulation that generates the correct output, they must once again delve into the code to check every line:

SA: ...I'm just trying to break it down
as much as possible so that we can
find any mistakes.

In this way, the group began to not only determine the correctness of lines of code that have been added/modified, but also began to examine the relationships *between* those lines.

For example, the group began by confirming the correctness of the form of one such line of code:

SA: Final momentum equals initial momentum
plus net force times delta t. True?
SC: Yeah...
SB: Yes.
SA: O.K. That's exactly what we have
here. So this is not the problem.
This is right.
SD: Yeah.

That is, Student A (*i*) read aloud and wrote down the line of code $\vec{p}_f = \vec{p}_i + \vec{F}_{\text{net}}\Delta t$ while the entire group confirmed on its correct form. This written line was then boxed, and was shortly followed up (*ii*) with a similar confirmation of the line $\vec{r}_f = \vec{r}_i + \vec{v}\Delta t$ that immediately prompted (*iii*) the confirmation of $\vec{v} = \vec{p}/m$. Thus, not only do we see the group determining the correctness of added/modified lines of code as in (*i*)–(*iii*), we further see confirmation with the links *between* those lines.

The confirmation of the link between the lines of code (*i*) and (*ii*), representing the incremental update of position and momentum in time, respectively, was evidenced not through the mere addition of the linking equation (*iii*) to the list of lines added, but further through the gestures exhibited by student A. Pointing at (*iii*), the \vec{v} in (*ii*), and the \vec{p}_f in (*i*), demonstrated that the group understood that without this linking equation (*iii*), the velocity used in (*ii*) would not reflect the time updated velocity by means of (*i*).

The group ran through these types of confirmations with FPPs rapidly over the span of a few minutes. Once the group had confirmed all the added/modified lines of code to their satisfaction, the discussion quieted down. The FPPs were winnowed from the discussion, and after a little more off-topic discussion we find them seeking help from the instructor:

SD: Maybe we should just stare at him
until he comes help us...

Suddenly, a haphazard change to the code:

SA: You know what, I'm gonna try
something.

where Student A changed the order of magnitude of the initial momentum a few times. This modification eventually resulted in a simulation that produced the correct output.

C. Bug resolution

At about 65 min into the problem, Student A changed the order of magnitude of the momentum one final time, which produced something *closer* to the output that they expected:

SA: Oh wait... Oh god...
SD: Is it working?

The satellite now elliptically orbited the Earth. This marks the end of the debugging phase and the beginning of the resolution phase – the bug had successfully been found and remedied. Given that the only line of code modified to produce this change was the initial momentum, they began to rethink the problem:

SD: I think that is the issue is that
we don't have the initial momentum...
SA: ...momentum correct?

That is to say, the group pursued the issue of determining the correct initial momentum with the added insight gained through debugging fundamentally correct VPython code.

V. DISCUSSION

To summarize, in analyzing this particular group, we first identified the two phases in time when the group had recognized and resolved a physics bug. We then necessarily identified the phase in-between as the process of physics debugging in P³, where the fundamentally correct code was taken from producing unexpected output to producing expected output. Given our assumption that the process of computer science debugging encourages desirable strategies, we then began to analyze this process of physics debugging further for strategies unique to P³.

Given the actions exhibited during the debugging phase, we can separate them into two distinct parts: a more strategic part and a less strategic part, as shown in Fig. 2. The group initially gave indication that they were working in a considerate, thorough, and consistent manner, which we classify as more strategic. This is contrasted by the later indications of more haphazard actions, which we classify as less strategic. These are the

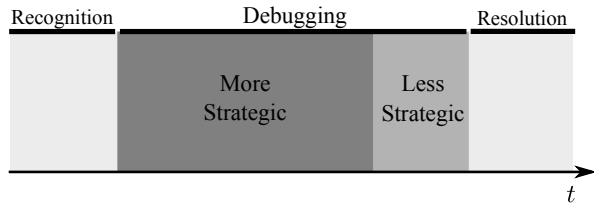


FIG. 2. Physics debugging phase consisting of more and less strategic strategies specific to our case study, bounded on either side by bug recognition and bug resolution.

two physics debugging strategies that, together, led to the resolution of the bug in this context.

The more strategic strategy was exhibited through the confirmation of individual FPPs as well as their relation to others. Not only did the group confirm through discussion, they simultaneously wrote, boxed, and referenced equations in the code – this helped to reduce the number of FPPs they needed to cognitively juggle at any given time [12]. This confirmation of FPPs through discussion presented a great learning opportunity for the entire group, where creative and conceptual differences could be jointly ironed-out. Accordingly, we tentatively refer to this strategy as **self-consistency**.

Although the resolution of the bug might not be tied directly to this self-consistency, that does not negate the learning opportunities afforded to the group along the way. Specifically, we saw the group double-checking every fundamental idea used and, possibly more importantly, the links between those ideas. Being physically self-consistent in this manner is a desired strategy in P³.

The less strategic strategy was exhibited during the haphazard changes to the initial momentum. These changes to the code that eventually resolved the bug, though one of the benefits of computation (i.e., the immediacy of feedback coupled with the undo function), could have been more thoughtful. A deeper understanding of the physics or computation could have tipped the group off to the fact that the initial momentum was too small.

Again, this does not negate the learning opportunities afforded to the group through this less strategic strategy, which resembles [13] that of “productive messing about.” Accordingly, we tentatively refer to this strategy as **play**.

Both of these strategies identified here, self-consistency and play, provided learning opportunities to the group which are bolstered by the computational nature of the task. In other words, the necessity of translating a collection of physical ideas into lines of code which must logically flow and the benefit of immediate visual display resulted in learning opportunities which might otherwise have been missed in an analytic task. More research is needed to dissect these learning opportunities and to deepen our understanding of the strategies themselves.

VI. CONCLUSION

This case study has described two strategies (one more and one less strategic) employed by a group of students in a physics course where students develop computational models using VPython while negotiating meaning of fundamental physics principles. These strategies arose through the group’s process of debugging a fundamentally correct program that modeled a geostationary orbit. The additional data we have collected around students’ use of computation is rich, and further research is needed to advance the depth and breadth of our understanding of the myriad of ways in which students might debug computational models in physics courses.

ACKNOWLEDGMENTS

The authors thank the members of PERL@MSU for their useful comments and suggestions at various stages of this work. We also thank Stuart Tessmer for his continued involvement in P³. This work is supported in part by the CREATE for STEM Institute at MSU.

-
- [1] DiSessa, Communications of the ACM **29** (1986).
 - [2] Shecker, Physics Education **28** (1993).
 - [3] N. Chonacky and D. Winch, American Association of Physics Teachers (2008).
 - [4] R. W. Chabay and B. A. Sherwood, American Association of Physics Teachers (2008).
 - [5] R. W. Chabay and B. A. Sherwood, American Journal of Physics **72** (2004).
 - [6] R. W. Chabay and B. A. Sherwood, American Journal of Physics **62** (1999).
 - [7] M. A. Kohlmyer, Ph.D. thesis, Carnegie Mellon University (2005).
 - [8] M. D. Caballero, M. A. Kohlmyer, and M. F. Schatz, Physics Review **8** (2012).
 - [9] R. McCauley, S. Fitzgerald, G. Lewandowski, L. Murphy, B. Simon, L. Thomas, and C. Zander, Computer Science Education **18**, 67 (2008).
 - [10] L. Murphy, G. Lewandowski, R. McCauley, B. Simon, L. Thomas, and C. Zander, in *ACM SIGCSE Bulletin* (ACM, 2008), vol. 40, pp. 163–167.
 - [11] B. Flyvbjerg, Qualitative Inquiry **12**, 219 (2006).
 - [12] E. F. Redish, *Teaching physics with the physics suite* (2003).
 - [13] N. S. Podolefsky, D. Rehn, and K. K. Perkins, in *Physics Education Research Conference* (2012).